Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nataša Vodopivec

Prenos meritev pametne zapestnice v enovito podatkovno bazo

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

prof. dr. Miha Mraz
MENTOR

doc. dr. Miha Moškon
SOMENTOR

Ljubljana, 2016

Tematika naloge:

Kandidatka naj v svojem delu analizira možnosti uporabe pametne zapestnice na področju varovanih stanovanj. Pri tem naj izbere najustreznejšo in jo uporabi pri izgradnji prototipa prenosa zajetih podatkov v enovito podatkovno bazo.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisana izjavljam, da sem avtorica dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelala samostojno pod mentorstvom prof. dr. Miha Mraza in somentorstvom doc. dr. Miha Moškona,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Nataša Vodopivec, Ljubljana, oktober 2016.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nataša Vodopivec

Prenos meritev pametne zapestnice v enovito podatkovno bazo

POVZETEK

V starosti se zaradi težav z gibanjem in skrbjo zase starejše osebe ponavadi preselijo v domove za starejše občane. Neprestana podvrženost okolju, kjer njihovi sobivanjci dnevno zbolevajo, pehajo in umirajo, starostnike lahko pripelje do padca morale in volje do življenja. Alternativo domovom za starejše občane predstavljajo varovana stanovanja, kjer je stopnja samostojnega življenja višja. Problem pa je, ker tu starostniki svoje težave težje sporočijo medicinskemu osebju. Tu se pojavlja tehnološka rešitev v obliki oskrbovanega stanovanja s senzorji. Meritve posameznih senzorskih naprav bi lahko združili na enotni platformi, kjer bi imeli celoten pregled nad zdravjem starostnika in stanjem stanovanja. V diplomski nalogi smo se lotili implementacije te rešitve. Meritve izbrane senzorske zapestnice smo zajeli z mobilno aplikacijo in jih nato poslali na računalnik, kjer smo jih zabeležili v podatkovno bazo. V taki obliki so podatki na voljo za nadaljno obdelavo bodisi v diagnostične namene ali pa za proženje izrednih dogodkov kot je nujen prihod medicinskega osebja.

Ključne besede: varovano stanovanje, internet stvari, pametne naprave, SensorTag, Android, brezžična senzorska omrežja.

University of Ljubljana
Faculty of Computer and Information Science

Nataša Vodopivec

Grouping data of a smart wearable device on a single platform

ABSTRACT

Due to difficulties in performing daily activities many elders seek help in retirement homes. But living in a closed space, being surrounded with equally old, sick or dying people, can lead to a higher rate of morbimortality. Alternative, offering a higher rate of mobility and independence, comes in the form of assisted living. However, in such homes, calling for aid in times of need can present difficulties. Development in technology brought the solution of Ambient Assisted Living. Collection of data from all sensors on a single platform would allow us to have full view of elder's health. The purpose of this thesis was an example implementation of such solution. Gathering data on a sensor device, we sent it to a mobile application, that served as a bridge forwarding data to a computer where it was stored in a database. As such, it could be used for further analysis either for diagnostic purposes or to raise alarm in need of urgent medical assistance.

Key words: ambient assisted living, Internet of Things, smart devices, SensorTag, Android, wireless sensor networks.

ZAHVALA

Zahvaljujem se družini in najljubšim sošolcem za vso podporo med študijem, mentorjema za mnogo pozitivne energije ob pisanju diplomske naloge in Cherry Choi za inspiracijo.

— Nataša Vodopivec, Ljubljana, oktober 2016.

KAZALO

1 Uvod

V večini razvitih držav se pojavlja trend naraščanja števila starejšega prebivalstva, za katerega se pričakuje, da bo v naslednjih treh desetletjih letih močno naraslo. Zaradi napredkov v znanosti in medicini se življenska doba ljudi povečuje, medtem ko natalitета v mnogih državah pada. Že leta 2000 je starost petine prebivalcev razvitega sveta presegala 65 let, napovedi pa kažejo da bo do leta 2050 ta količina zrasla na eno tretjino. V manj razvitih predelih sveta bo količina iz osmih procentov leta 2000, do leta 2050 zrasla na dvajset procentov. Do leta 2050 bo število starejših oseb prvič večje od števila mladih. V starosti se pogosto začnejo pojavljati zdravstvene težave in kronične bolezni, kjer so med pogostejšimi demenca, rak, osteoporoza, dihalne težave, artritis, srčna obolenja, diabetes in druge. Zaradi pogostosti bolezni pri starostnikih kaže, da jih bo leta 2030 40% potrebovalo pomoč družinskih članov ali zdravstvenega osebja pri opravilih vsakodnevnega življenja [1].

Samostojnost pozitivno pripomore k človekovemu dobremu počutju skozi njegovo celotno življenje. V starosti je slednja zaradi težav z gibanjem in skrbjo zase pogosto okrnjena, zato se starejše osebe ponavadi preselijo v domove za starejše občane.

Institucializirano bivanje poleg koristi prinaša tudi nekaj slabosti. V domovih za starejše so starostniki podrejeni nadzoru medicinskega osebja, kar privede do izgube zasebnega prostora in mnogo dejavnosti ne morejo več opravljati sami, tudi če bi si jih želeli. Znajdejo se v družbi samih starejših oseb z več ali manj podobnimi zdravstvenimi težavami kot jih imajo tudi sami. Neprestana podvrženost okolju, kjer njihovi sobivanjci dnevno zbolevajo, pehajo ali umirajo lahko pripelje do padca morale in volje do življenja. Njihov vsakdan je uokvirjen v to kar ustanova, kjer živijo, ponuja in hitro se lahko zgodi, da izgubijo stik z življenjem izven zaprtega okolja doma.

Poleg tega izguba mobilnosti neposredno privede do delne ali popolne izgube neodvisnosti, kar tudi vpliva na starostnikovo moralo in lahko privede celo do višje stopnje umrljivosti [16]. Zato je očitno, da preventivni posegi za ohranjanje ali obnavljanje mobilnosti starejših lahko izboljšajo dolžino in kakovost njihovega življenja.

1.1 Varovana stanovanja

Alternativo domovom za starejše osebe predstavlja selitev v varovana stanovanja. V teh je stopnja samostojnega življenja višja, cena pa je slabši vpogled dežurnega medicinskega osebja nad zdravjem varovancev, ki morajo če potrebujejo nujno pomoč, uporabiti klicno napravo. Klicna naprava je ponavadi implementirana v obliki gumbov na katere mora starostnik pritisniti, če potrebuje pomoč, kar pa se mnogokrat izkaže za nepraktično rešitev. Ena pogostejših nesreč, ki se zgodi starostnikom je padec, kjer starostnik lahko izgubi zavest, zaradi bolečin ali šoka ne more vstati ali pa pozabi na gumb in ga tako ne more uporabiti. Zato je potreben sistem, kjer starostniki ob nesreči skrbnikov in medicinskega osebja ne potrebujejo obvestiti sami, temveč se to zgodi avtomatsko.

Tu se pojavlja tehnološka rešitev v obliki oskrbovanega stanovanja s senzorji *AAL* (ang. *Ambient Assisted Living*). Ideja je starostniku zagotoviti podaljšano samooskrbo, skrbeti za njegovo varnost, zagotoviti njegovo zadovoljstvo, razbremeniti svojce, skrbnike ter medicinsko osebje in predvsem časovno odložiti prehod starejših v institucialno obliko bivanja ter zmanjšati potrebo po stalnem medicinskem nadzoru z uporabo različnih senzorskih naprav.

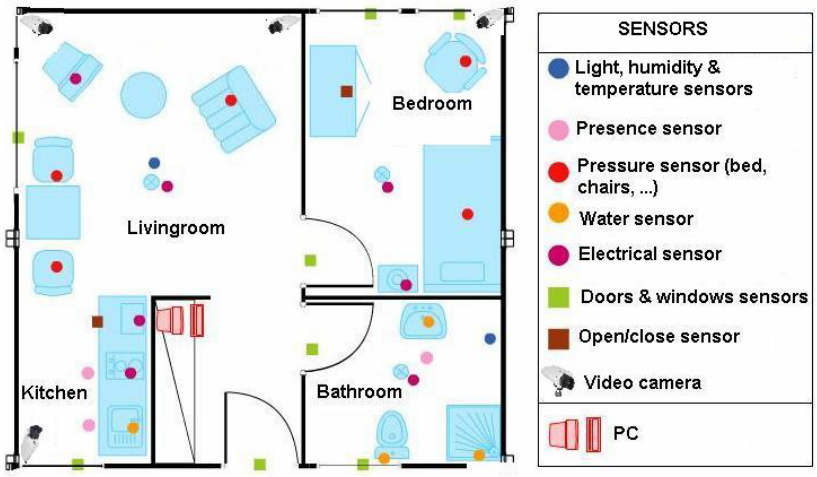
1.2 Varovano stanovanje s senzorji

Glavne funkcije naprav v varovanem stanovanju s senzorji so zdravstveno opazovanje starostnika in obveščanje skrbnikov ob nenavadnih dogodkih (na primer preko telefonskega sporočila, klica, spletne pošte, alarma), možnost zbiranja podakov v zdravstveni karton, skrb za pravilno prehrano in gibanje starostnika, varnost ter možnost zdravstvenih opomnikov [19]. Eden glavnih pokazateljev fizičnih in psihičnih sposobnosti starostnika je sposobnost samostojnega opravljanja vsakodnevnih življenskih funkcij ADL (ang. *Activities of Daily Living*) kot so kuhanje in uporaba kopalnice. Raziskave so pokazale, da imajo starostniki pogosto težave pri tuširanju, kjer ali zelo dolgo stojijo pod tušem, ker pozabijo kje so, ali pa pozabijo uporabiti milo. Če v kopalnici pozabijo zapreti pipo, lahko voda zalije tla, kar poveča možnost zdrsa in padca. Pogosto se zbudijo sredi noči in tavajo naokoli. Včasih zamenjajo osebno lastnino in spijo na napačni postelji ali oblečejo obleke druge osebe [15]. Zato morajo biti senzorji v varovanem stanovanju sposobni opazovanja tudi vsakodnevnih življenskih opravil starostnika.

Za celotno senzorsko pokritost stanovanja se uporablja mnogo različnih senzorjev. To so lahko video kamere, senzorji gibanja, senzorji pritiska, senzorji s tehnologijo GPS, električni senzorji, senzorji svetlobe in temperature, senzorji zaznavanja vibracij in pretoka vode, RFID značke in drugi. Pomembno je, da je kljub množici senzorjev poskrbljeno za zasebnost starostnika. Primer pokritosti stanovanja je prikazan na sliki 1.1. Senzorji med sabo ponavadi komunicirajo preko tehnologij WiFi, Bluetooth, ZigBee in drugih.

Na tržišču že obstaja mnogo naprav, ki služijo namenu opremljanja senzorskih varovanih stanovanj, a vsak proizvajalec senzorskih naprav svoje podatke oddaja, sprejema in obdeluje na samosvoj način, zato je težko imeti pregled nad vsemi prejetimi podatki, prevelike količine podatkov in meritev pa lahko tako postanejo neuporabne. Prav tako se vsaka naprava povezuje z drugačno platformo ali pa ponuja drugačen način sporočanja svojih meritev skrbnikom, kar jim vloge skrbnika ne olajša do te mere kot bi jo lahko. Uporabna tehnološka rešitev bi bila združitev meritev vseh povezanih naprav na enotni platformi, kjer bi se tako razpolagalo s celotnim spektrom podatkov, do katerih bi se potem dostopalo glede na specifično potrebo in se jih uporabljalo naprej v želeni obliki.

Pri gradnji platforme igra ključno vlogo njena skalabilnost in preprostost uporabe. Sistem mora:



Slika 1.1: Shema s senzorji opremljenega varovanega stanovanja [21].

- biti neodvisen od komunikacijske tehnologije,
- omogočati uporabo senzorjev različnih proizvajalcev,
- biti prilagodljiv uporabnikom z različnimi potebami,
- omogočati preprosto dodajanje novih senzorjev in
- imeti nizko porabo energije [19].

Potrebno je omogočiti dovolj nizko ceno vzpostavitve celotnega sistema, da bi se starostnikom oprema senzorskega varovanega stanovanja splačala in bila bolj ugodna od selitve v dom za ostarele.

1.3 Zapestnica

Kot senzor za pridobivanje osnovnih zdravstvenih podatkov o pacientu so trenutno na trgu najbolj razširjene zapestnice. So majhne in prenosljive, ter pacienta pri uporabi najmanj ovirajo. Na trgu obstaja tudi nekaj zapestnic, ki spodbujajo razvijalce, da z njimi naredijo lastne programske rešitve. V pričujočem delu smo za razvoj rešitve uporabili zapestnici *AngelSensor* in *SensorTag*.

1.4 Pregled dela

V diplomski nalogi bi radi izdelali rešitev, kjer bi več senzorskih naprav svoje meritve pošiljalo na skupno točko, ki bi delovala kot koncentrator podatkov vezanih na oskrbovano stanovanje in oskrbovanca. Naprave bi delovale znotraj zaprtih mej stanovanja, njihove meritve pa bi bile vidne le koncentratorju, ki bi deloval kot vstopna točka za zunanji svet. Pilotno smo delali le z eno napravo, naš končni cilj pa je podatke iz naprave prikazati v podatkovni bazi na osebnem računalniku, ki naj bi v pilotni fazi prevzel nalogo koncentratorja.

Uporabljene tehnologije so naštete v drugem poglavju. Opisani so uporabljeni senzorji s katerimi smo zajemali podatke. Nato je obrazloženo prenašanje podatkov z dvema različnima protokoloma in okolja, ki so za to potrebna. Na koncu je opisano kako podatke shranjujemo.

V tretjem poglavju je na začetku prikazana shema naše rešitve. Sledi podrobna obrazložitev vsakega koraka implemetacije. V zaključku podamo naše sklepe in možnost za nadaljnje delo.

2 Izbira tehnologij

V pričujočem poglavju predstavimo izbrane tehnologije za izgradnjo rešitve in njihovo izbiro utemeljimo.

2.1 Angel Sensor

Naša prva želja je bila uporaba zapestnice *Angel Sensor* podjetja Seraphim Sense [2].

Slika 2.1: Zapestnica AngelSensor [2].

Angel Sensor je pametna zapestnica, ki je zaradi svojih odprtokodnih rešitev na-

menjena predvsem razvijalcem. Izbrali smo jo, ker smo jo videli kot rešitev enega od mogočih scenarijev testne implementacije avtomatiziranega varovanega stanovanja. Ponuja namreč neomejen dostop do svojih senzorjev in uporabniku vrača neobdelane podatke (ang. *raw data*) z uporabo tehnologije BLE (ang. *Bluetooth Low Energy*). Razvita je bila z namenom, da lahko vsak razvijalec z njeno uporabo razvije svoje aplikacije. Je precej univerzalna, saj preko svojega SDK-ja (ang. *Software Development Kit*) podpira tako razvoj na iOS kot tudi na Android okoljih. Ker je zapestnica namenjena nošenju okoli zapestja je bil scenarij, ki smo ga imeli v mislih, opazovanje starostnikovega srčnega utripa in temperature, saj naj bi senzorji te meritve omogočali.

Na spletni strani proizvajalca [2] so naštete vse meritve, katere naj bi bila zapestnica sposobna izvajati. Pri nekaterih funkcionalnostih je označeno, da so še v razvoju(*). Zapestnica meri naslednje podatke [3]:

- srčni utrip z uporabo PPG (ang. *photoplethysmography*),
- količino kisika v krvi*,
- temperaturo kože,
- porabljene kalorije*,
- število prehojenih korakov,
- kvaliteto spanja iz analize premikanja,
- gibanje zapestnice z uporabo giroskopa (ang. *gyroscope*).

Zapestnica svoje stanje uporabniku sporoča z utripanjem lučk, s tresenjem in preprostimi toni.

Dostop do meritev, ki jih zapestnica opravlja, je omogočen z vnaprej pripravljenim SDK-jem. Deluje preko dveh protokolov. Prvi je standardna Bluetooth BLE storitev, drugi pa je zapestničin lasten.

Protokoli Bluetooth omogočajo meritve srčnega utripa (ang. *heart rate*), temperature kože (ang. *health thermometer*), stanja baterije (ang. *battery service*) in osnovnih informacij o napravi (ang. *device information*).

Protokoli Seraphim Sense preko pospeškometra in giroskopa zaznavajo premikanje zapestnice, s čimer merijo število opravljenih korakov; ugotavljajo kje na telesu se zapestnica nahaja in zaznavajo padce (kar pa še ni implemenitrano). Eksperimentalno obstaja tudi izris meritev pospeškometra in svetlobnih utripov v obliki grafa.

Glede na uporabnost zapestnice, ki je bila obljubljena na spletni strani smo bili nad Angel Sensorjem zelo navdušeni. Ko smo ga prejeli pa so se začele pojavljati težave. Ena od zapestnic je popolnoma prenehala delovati, druga pa ima težave s povezovanjem na telefone preko Bluetooth povezave. Od telefonov več različnih proizvajalcev z enakimi verzijami sistema Android jo je zaznal le eden. V dokumentaciji nismo uspeli najti rešitve za nobenega od problemov, saj je zelo skopa s svojimi nasveti. Ker je produkt precej nov, rešitve nismo našli niti drugje na spletu, zato smo se uporabi zapestnice za svoje potrebe odpovedali.

2.2 CC2650 SensorTag

Zaradi tehničnih problemov s prvo izbrano tehnologijo smo nato poizkusili še s senzorjem *SimpleLink CC2650 SensorTag* (v nadaljevanju *SensorTag*) podjetja Texas Instrumets [4].

Slika 2.2: Texas Instruments CC2650 SensorTag [4].

Tako kot je obljubljala zapestnica Angel Sensor, tudi SensorTag ponuja neomejen dostop do svojih senzorjev in uporabniku vrača neobdelane podatke z uporabo tehnologije BLE. Ta omogoča razvoj aplikacij na različnih arhitekturah (iOS ali Android). V nasprotju z napravo AngelSensor je SensorTag, poleg za razvoj mobilnih aplikacij, odprt tudi za razvoj lastnih aplikacij, ki tečejo na sami napravi.

Strojna oprema

SensorTag je narejen na osnovi brezžičnega mikrokrmilnika SimpleLink CC2650, ki predstavlja nizkoenergijsko rešitev z vgrajeno podporo različnim brezžičnim komunikacijskim standardom na eni sami silicijevi rezini SoC (ang. *System on Chip*). SimpleLink CC2650 vsbuje:

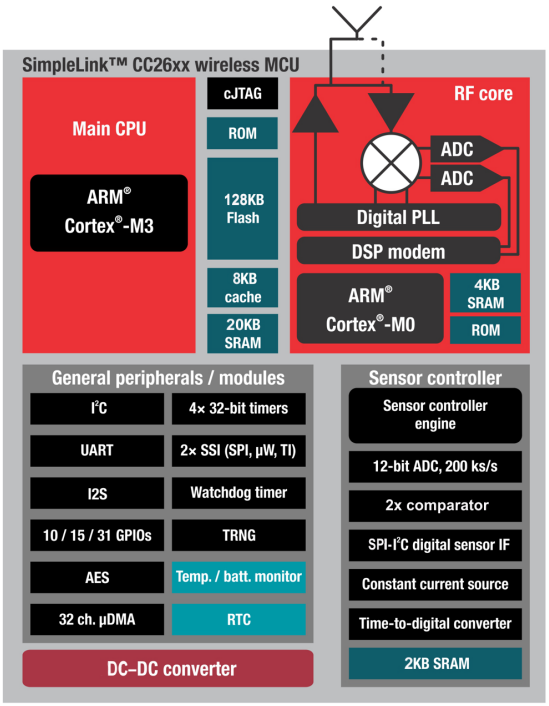
- 2.4GHz radijskofrekvenčni sprejemnik-oddajnik (ang. *transciever*)s podporo Bluetooth Low Energy (BLE) in IEEE 80215.4,
- procesor ARM®Cortex®-M3 s frekvenco 48MHz,
- 128 KB programabilnega spomina,
- 20 KB statičnega spomina (SRAM) in
- obsežen nabor vmesnikov.

Vezje temelji na mikroprocesorskem jedru ARM®Cortex®-M3 (CM3), ki skrbi za aplikacijski nivo sklada BLE protokola in neodvisnem radijskem mikroprocesorskem jedru ARM®Cortex®-M0 (CM0), ki deluje kot vmesnik radijski strojni opremi ter kot prevajalnik ukazov procesorja CM3 v manjše enote, ki se jih lahko pošlje preko radijskih valov. CM0 pogosto lahko deluje avtonomno, kar sprosti CM3. Na 32-bitnem ARM®Cortex®-M3 procesorju temelji procesna moč vezja.

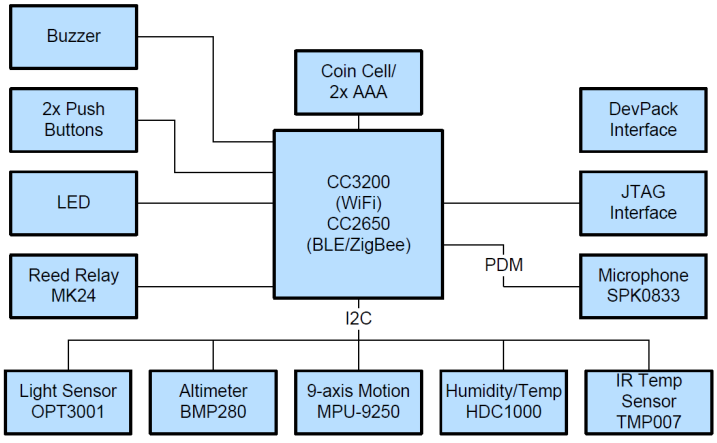
Najpomembnejši vmesnik je nizkoenergijski nadzorni sistem za senzorje (ang. *sensor controller*), ki deluje kot vmesnik za vse povezane zunanje senzorje. Njegova druga naloga je avtonomno zbiranje analognih in digitalnih podatkov, medtem ko je preostanek sistema v načinu spanja. Shematski prikaz mikrokrmilnika CC2650 je na sliki 2.3.

Vezje CC2650 podpira več standardnih komunikacijskih protokolov, kot so BLE, ZigBee in 6LoWPAN (ang. *IPv6 over low-power wireless personal area networks*).

SensorTag vsebuje 10 nizkoenergijskih senzorjev, ki merijo ambientalno svetlobo, infrardečo temperaturo, ambientalno temperaturo, gravitacijski pospešek, kotne pospeške, magnetno polje in vlago. Vključuje še mikrofon in magnetni senzor. Shematski prikaz naprave SensorTag je na sliki 2.4.



Slika 2.3: Zgradba mikrokrmilnika CC2650. [17].



Slika 2.4: Zgradba naprave SensorTag. [17].

Osnovni SensorTag lahko nadgradimo z eno od treh razširitev. Nanj lahko dodamo preprost ekran in ga tako spremenimo na primer v uro ali vremensko postajo, lahko ga opremimo z žarnico v kateri so 4 večbarvne LED diode ali pa ga razširimo z razvijalskim dodatkom z razhroščevalnikom, ki nam omogoča programiranje sistemske programske opreme (ang. *firmware*).

Sistemska programska oprema

Razširitev naprave SensorTag z razvijalskim dodatkom nam omogoča razvoj sistemske programske opreme na enem od naslednjih dveh okolij:

- *IAR Embedded Workbench for ARM* [5],
- *Code Composter Studio* (CSS) [6].

CSS je razvil isti proizvajalec kot SensorTag, zaradi česar je priporočeno razvijalsko orodje. Namesto programa, ki ga moramo na svoj računalnik pred uporabo naložiti, lahko za manj kompleksne projekte uporabljamo spletno verzijo razvojnega okolja CCS. Projekt lahko razvijemo, zgradimo in naložimo na flash spomin v napravi SensorTag neposredno iz spletnega vmesnika. Razvijamo v jeziku C oziroma C++.

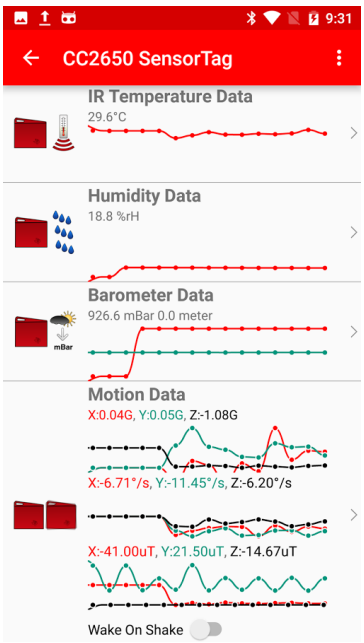
V našem delu smo uporabili sistemsko programsko opremo, ki je bila na napravi že nameščena in omogoča vklapljanje in izklapljanje vseh senzorjev, pošiljanje prebranih vrednosti preko BLE in nastavljanje periode branja ter pošiljanja podatkov. Komunikacija s sprejemno BLE napravo pri tem poteka preko mobilne platforme z uporabo iOS ali Android aplikacije.

Programska oprema na mobilni platformi

Mobilna aplikacija je za iOS v spletni trgovini aplikacij na voljo pod imenom *TI SensorTag*, za Android pod imenom *Simplelink SensorTag*, poleg tega pa je odprta tudi za razvoj. Izvorno kodo aplikacije lahko najdemo na spletni strani proizvajalca, kjer se imenuje *BLE SensorTag*.

Prvi zaslon aplikacije je namenjen povezovanju na BLE napravo. S pritiskom na gumb z napisom *Scan* (išči) odkrijemo BLE naprave v stanju oddajanja signala, ki se prikažejo na seznamu. Izberemo eno od naprav in se nanjo povežemo s pritiskom na gumb z napisom *Connect* (poveži). Po uspešni povezavi se na zaslonu prikaže seznam, ki za vsako

od meritev, torej meritev vlažnosti zraka, zračnega pritiska, temperature prostora, temperature objekta, količine svetlobe in 9-osnega senzorja za zaznavanje gibanja, v svoji vrstici prikazuje številčno vrednost trenutnega podatka in izrisuje graf. Posamezno meritev lahko izklopimo ali pa spremenimo frekvenco zajemanja podatkov. Seznam na vrhu vsebuje tudi polje z informacijami o povezanem SensorTagu kot so številka njegovega modela, datum zadnje posodobitve strojne opreme in druge. Uporabniški vmesnik Android aplikacije prikazuje slika 2.5.



Slika 2.5: Izgled aplikacije TI SensorTag [4].

Delovanje naprave

Ob prižigu SensorTag oddaja svoj signal, da ga sprejemne naprave lahko odkrijejo in z njim vzpostavijo povezavo. Da bolje ohranja baterijo, signal oddaja le približno dve minuti, saj je oddajanje energijsko visoko potraten način delovanja. BLE oddajnikom omogoča povezavo le na en sprejemnik, zato je SensorTag ob vsakem trenutku lahko povezan le z eno napravo. Ko sprejemna naprava vzpostavi povezavo, mora na SensorTagu najprej prebrati katere storitve in karakteristike so omogočene. Nato lahko omogoči senzorje, ki pripadajo želenim storitvam in se naroči na njihove notifikacije, da lahko iz njih bere podatke in s pisanjem vanje nastavlja njihove nastavitve. BLE protokol omogoča

povezavo med katerikolim oddajnikom in sprejemnikom dokler oba podpirata BLE, zato se SensorTag poleg na pametne telefone in tablice lahko poveže tudi na druge naprave, ki imajo ustrezno implementacijo iskanja BLE oddajnikov. Postopek povezovanja naprav z BLE je podrobneje opisan v razdelku 2.4.2.

SensorTag glede na vgrajene senzorje podpira naslednje profile (glej razdelek 2.4.1):

- profil vlažnosti (ang. *Humidity Profile*),
- barometer (ang. *Barometric Pressure Profile*),
- profil ambientalne svetlobe (ang. *Ambient Light Profile*),
- profil infrardeče temperature in temperture okolja (ang. *Infrared and Ambient Temperature Profile*),
- profil za 9-osni senzor za zaznavanje gibanja (ang. *9-axis Motion Tracking Device*), ki vključuje pospeškometer (ang. *Accelerometer*), giroskop in merilec magnetnega polja (ang. *Magnetometer*),
- profil gumba za dodatne uporabniške funkcije (ang. *Key Service*),
- posodabljanje programske opreme preko spletne povezave OAD (ang. *Over the Air Download*),
- objavljanje meritev na oblačno storitev.

Vsak profil napravi omogoča, da bralnim napravam izpostavlja stanje in meritve pripadajočih senzorjev, nastavitve njihovih karakteristik (lahko so omogočene ali onemogočene) in nastavitve intervala branja meritev.

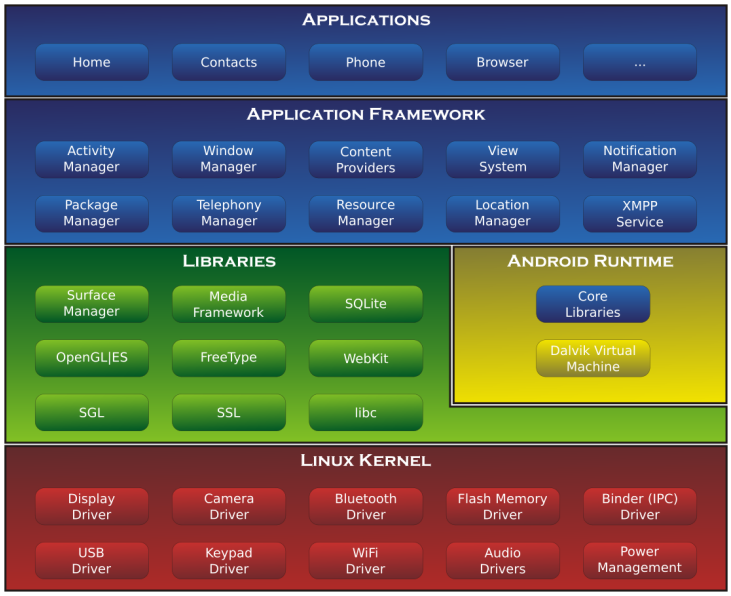
Ker SensorTag podatke pošilja v neobdelani obliki, jih je pred uporabo v aplikacijah potrebno umeriti. Navodila za kalibracijo nekaterih senzorjev ponuja Texas Instruments v svojem priročniku [7], za bolj natančno kalibracijo drugih pa priporočajo sodelovanje direktno s proizvajalci teh senzorjev.

Podrobne informacije o načinu povezave in uporabe vsakega senzorja, ter tabele GATT atributov (glej razdelek 2.4.1) z vsemi potrebnimi podatki, se nahajajo na spletni strani proizvajalca [4].

2.3 Android

Android je odprtokodni operacijski sistem, ki temelji na sistemu Linux. Sistem so leta 2003 začeli razvijati v Kaliforniji, leta 2005 pa ga je kupilo in 2007 spravilo na trg podjetje Google. Od takrat je bil deležen mnogih posodobitev, z vsako pa nudi podporo večim napravam. V glavni meri je namenjen mobilnim napravam z zasloni na dotik, kot so pametni telefoni in tablice.

Vsak Android sistem je sestavljen iz petih osnovnih komponent: aplikacije, aplikacijskega ogrodja, knjižnic, zagonskega okolja in jedra s sistemom Linux. Zgradba sistema je prikazana na sliki 2.6.



Slika 2.6: Zgradba Android sistema [8].

Zaradi večje varnosti je Android operacijski sistem zasnovan kot večuporabniški Linux sistem, kjer je vsaka aplikacija svoj uporabnik. Teče v svojem lastnem Linux procesu, kar omogoča, da so aplikacije ločene druga od druge, ter da se jih lahko neodvisno ustvarja in končuje. Aplikaciji se ob zagonu dodeli identifikacijska številka preko katere do nje lahko dostopajo druge aplikacije in preko katere lahko zaprosi sistem, da ji odobri dovoljenja za dostop do uporabniških datotek in podatkov kot so sporočila, imenik, galerija slik, Bluetooth povezava in druge. Priporočeno okolje za razvoj Android mobilnih aplikacij je

Android Studio.

2.3.1 Sestava Android aplikacije

Android aplikacije so običajno napisane v objektnem programskem jeziku Java. Pišemo jih s pomočjo zbirke Android SDK, ki ponuja širok izbor razvojnih orodij kot so razhroščevalnik (ang. *debugger*), knjižnice, posnemovalnik realnih naprav (ang. *emulator*), dokumentacija, primeri kode in obrazložena navodila (ang. *tutorials*). Orodja SDK kodo prevedejo v paketno datoteko APK (ang. *Android Package*) s končnico `.apk`. To datoteko Android naprave uporabljajo za izvajanje programa, saj vsebuje vse lastnosti aplikacije.

Komponente

Vsaka aplikacija je sestavljena iz štirih različnih komponent, ki so med seboj neodvisne, vsaka pa predstavlja zaključeno entiteto:

- Aktivnost (ang. *Activity*),
- Storitev (ang. *Service*),
- Upravitelj vsebine (ang. *Content Provider*),
- Sprejemnik (ang. *Broadcast Receiver*).

Aktivnost predstavlja vizualni uporabniški vmesnik. Aplikacija je ponavadi sestavljena iz več aktivnosti od katerih vsaka naenkrat zasede celoten zaslon naprave. Pri aplikaciji spletne pošte je na primer ena aktivnost namenjena prikazu prejete pošte, druga aktivnost pisanju novega sporočila, tretja pa branju sporočil. Čeprav vse te aktivnosti sestavljajo celotno aplikacijo, lahko teče tudi vsaka posebej, saj so druga od druge neodvisne. Aktivna je lahko le ena naenkrat, zažene pa jo lahko tudi katera od drugih aplikacij. Ko uporabnik v galeriji slik (ki je tudi svoja aplikacija) izbere možnost, da bi sliko rad delil s prijatelji, galerija zažene aktivnost aplikacije spletne pošte, ki je namenjena pisanju novega sporočila.

Storitev je komponenta brez uporabniškega vmesnika, ki teče v ozadju in izvaja dalj časa trajajoče opravilo, za katerega nočemo, da moti uporabnike s prikazom čez cel zaslon. Primer storitve je predvajalnik glasbe.

Ponudnik vsebine poskrbi za podatke aplikacije, ki so deljivi z drugimi aplikacijami. Podatke se lahko spravi na datotečni sistem, v SQLite podatkovno bazo, na splet ali katerokoli drugo končno hranilno lokacijo, do katere aplikacija lahko dostopa. Ponudnik vsebine drugim aplikacijam s potrebnimi pooblastili kasneje omogoča, da do teh podatkov dostopajo ali jih celo spreminjajo.

Sprejemnik je komponenta, ki sprejema in reagira na razpršena sistemska obvestila (ang. *system-wide broadcast announcements*). Veliko obvestil namreč izvira iz samega sistema. Sporočijo lahko na primer, da se je ugasnil zaslon, da zmanjkuje baterije ali, da je uporabnik naredil fotografijo. Aplikacija lahko obvestilo tudi pošlje. Sprejemniki nimajo uporabniškega vmesnika, uporabnika pa o dogodku lahko obvestijo preko notifikacije v statusni vrstici.

Datoteka manifest

Sistem mora vedeti katere komponente obstajajo, preden jih lahko uporabi. Aplikacija svoje komponente prijavi v *manifest datoteki*, ki je vključena v korenski imenik paketa APK skupaj z vso kodo aplikacije. Poleg komponent opisuje tudi katere knjižnice, dovoljenja, strojno (ang. *hardware*) in programsko (ang. *software*) opremo aplikacija potrebuje za pravilno delovanje.

2.3.2 Razvojno okolje Android Studio

Android Studio je uradno integrirano razvojno okolje IDE (ang. *Integrated Development Environment*) za razvoj Android aplikacij, ki temelji na programskem okolju JetBrains IntelliJ IDE [9]. Poleg močnega urejevalnika kode in razvijalnih orodij, Android Studio ponuja še mnogo drugih vtičnikov prilagojenih prav za razvoj androidnih aplikacij, zato je bila njegova izbira samoumevna.

2.4 Bluetooth Low Energy

Nizkoenergijski Bluetooth ali pametni Bluetooth (ang. *Bluetooth Smart*) je različica klasičnega Bluetootha z optimizirano porabo energije pri brezžičnem prenosu preko radijskih valov. Domet radijskega prenosa se giblje med 10 – 30 metri, s pomočjo merjenja moči prejetega signala RSSI (ang. *Received Signal Strength Indicator*) pa je mogoče ugotoviti približno kako daleč se nahaja naprava, ki BLE signal oddaja. BLE je od omrežja

neodvisna radijska tehnologija, kar pomeni, da za svoje delovanje ne potrebuje Interneta ali WiFi-ja. Naprave, ki jo podpirajo so ponavadi majhne in imajo baterijo, ki lahko zdrži več let. Zaradi vseh teh značilnosti je ena glavnih tehnologij, ki omogočajo Internet stvari (ang. *Internet of Things — IoT*) in se zadnje čase pojavlja v vse več potrošniških izdelkih, od igrač do športne opreme, gospodinjskih aparatov in drugih tehnoloških rešitev. Android je BLE začel podpirati od verzije 4.3 naprej.

Pri BLE komunikaciji imajo sodelujoče naprave več vrst vlog. So ali v relaciji gospodar-suženj (ang. *master-slave*), pri čemer ima vlogo sužnja naprava, ki signale oddaja (ang. *advertising*), zato jo imenujemo tudi oddajnik in vlogo gospodarja naprava, ki signale išče in sprejema (ang. *scanning*), ki jo imenujemo tudi sprejemnik ali pa v relaciji strežnik-odjemalec, kjer je strežnik (ang. *server*) naprava, ki podatke ima, odjemalec (ang. *client*) pa naprava, ki do njih dostopa in jih bere ali spreminja. Ponavadi ima naprava, ki ima vlogo sužnja, hkrati tudi vlogo strežnika, naprava z vlogo lastnika pa je tudi odjemalec, kar pa ni nujno.

Vlogo strežnika imajo najpogosteje navigacijske naprave, merilci srčnega utripa, prenosni termometri in drugi podobni pripomočki, vlogo odjemalca pa mobilne naprave.

Strežnik in odjemalec se med seboj pogovarjata s šestimi osnovnimi klici:

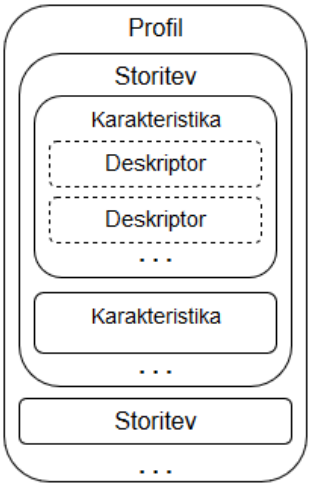
1. odjemalec *zaprosi* (ang. *request*) strežnik, naj nekaj naredi,
2. ob prošnji odjemalca, strežnik *odgovori* (ang. *response*),
3. odjemalec *ukaže* (ang. *command*) strežniku, naj nekaj naredi, pri čemer strežnikov odgovor ni potreben,
4. z *znakom* (ang. *indication*) strežnik odjemalcu sporoči, da je določen atribut dobil vrednost,
5. odjemalec odgovori s *potrditvijo prejema znaka* (ang. *confirmation*),
6. z *obvestilom ali notifikacijo* (ang. *notification*) strežnik odjemalcu sporoči enako kot z znakom, le da ne pričakuje potrditve prejema sporočila.

V naši aplikaciji, opisani v razdelku 3.2.2, je pogosta uporaba notifikacij.

2.4.1 Protokol GATT

Pravila za vzpostavljanje povezave in nadaljno komunikacijo preko BLE povezave določa *protokol GATT* (ang. *Generic Attribute Profile*), ki definira, da se podatke prenaša v obliki atributov. Ti so najmanjše zaokrožene podatkovne enote, ki jih definirata GATT in ATT (ang. *Attribute Profile* je protokol, na katerem sedi protokol GATT) in se jih lahko *naslavlja*. Vsak atribut ima 128-bitno unikatno identifikacijsko številko imenovano UUID (ang. *universally unique identifier*), ki mu z veliko verjetnostjo zagotavlja, da bo globalno unikatno označen. Zaradi hitrejšega prenosa je Bluetooth definiral zbirko standardnih UUID številk, ki so le 16-bitne, če pa uporabnik za svoje atribute želi lasten UUID, mora uporabiti vseh 128-bitov. Ponavadi so atributi locirani na strežniku v obliki tabele, v kateri se nahajajo tudi vsi njihovi opisni podatki (ang. *meta data*) kot so unikatna lokalna številka (ang. *handle*), tip in vrednost.

Ker GATT in ATT delujeta le s prenašanjem atributov, morajo biti vsi podatki zbrani v te podatkovne enote. Vsak atribut pa je sestavljen še iz treh konceptov. *Storitve* so sestavljene iz *katakteristik*, ki lahko vsebujejo *deskriptorje*. Zgradba atributa je prikazana na sliki 2.7.



Slika 2.7: Prikaz zgradbe GATT profila in atributov.

Storitve (ang. *services*) so skupine karakteristik, ki skupaj sodelujejo pri opravljanju specifične funkcije. Storitev za merjenje srčnega utripa je recimo lahko sestavljena iz naslednjih treh karakteristik: meritev srčnega utripa, lokacije merilnika na telesu, kontrole

srčnega utripa [11].

Karakteristika (ang. *characteristic*) je dejanski podatek, ki se prenaša med oddajno in sprejemno BLE napravo.

Na najnižjem nivoju govorimo o *desktriptorijih* (ang. *descriptor*), ki so dodatne informacije o karakteristiki. Najpogosteje uporabljen deskriptor je CCCD (ang. *Client Characteristic Configuration Descriptor*), s katerim odjemalec strežniku sporoči ali ima omogočeno sprejemanje notifikacij.

BLE protokolu GATT doda še en pojem. Vsaka naprava ima lahko enega ali več *profilov* (ang. *Profiles*), ki so pravzaprav le zbirke več storitev (ang. *Service*), ki združene pokrivajo kakšnega od primerov uporabe. Profil srčnega utripa je na primer sestavljen iz storitve za merjenje srčnega utripa in storitve z informacijami o napravi.

2.4.2 Postopek povezave med Androidom in BLE napravo

BLE povezavo se v Androidu ponavadi vzpostavi v naslednjih korakih:

1. nastavitev BLE dovoljenj v aplikaciji,
2. preverjanje, če odjemalec podpira BLE,
3. iskanje naprav, ki oddajajo želene storitve,
4. povezovanje na napravo,
5. iskanje storitev,
6. iskanje karakteristik,
7. prijava na prejem notifikacij za karakteristike,
8. zapiranje povezave.

2.5 SQLite

SQLite je relacijska podatkovna baza, implementirana v programskem jeziku C. Za svoje delovanje ne potrebuje zunanjih knjižnic in ni vezana na noben operacijski sistem, zato jo je preprosto uporabljati na množici naprav. Je vgrajena podatkovna baza SQL. V nasprotju z večino drugih podatkovnih baz za svoje delovanje ne potrebuje ločenega strežniškega procesa. Program, ki jo uporablja, bere in piše neposredno v datoteko na

trdem disku, kar pomeni, da jo lahko uporablja vsak program, ki ima dostop do diska. Ker ne potrebuje strežnika, je pred uporabo ni potrebno namestiti, omogoča zagon brez poznavanja naprednih programov za upravljanje s podatkovnimi bazami ter ne potrebuje administratorja, ki bi uporabnikom dodeljeval pravice za uporabo.

Celotna baza z vsemi tabelami je vključena v eno datoteko na disku. Ta datoteka je prenosljiva tako med različnimi operacijskimi sistemi kot tudi med 32- in 64-bitnimi sistemi in arhitekturami tankega in debelega konca (ang. *little-endian and big-endian*).

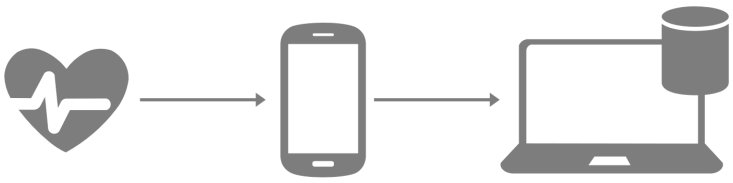
SQLite transakcije podpirajo koncept ACID, kar pomeni, da so atomične (ang. *Atomic*), konsistentne (ang. *Consistent*), izolirane (ang. *Isolated*) in trajnostne (ang. *Durable*). Preprosteje povedano to pomeni, da se vse spremembe, ki se zgodijo med eno transakcijo ali popolnoma zapišejo v bazo ali pa se sploh ne, tudi ob napakah operacijskega sistema ali izpadih elektrike [13].

3 Tehnična rešitev

V pričujočem poglavju predstavimo tehnično rešitev zajema, prenosa in hrambe podatkov, zajetih z zapestnico.

3.1 Shema rešitve

Z uporabo v prejšnjem razdelku naštetih tehnologij smo se lotili realizacije rešitve našega problema. Izgled končnega sistema je upodobljen na sliki 3.1. SensorTag iz okolja s svojimi senzorji prebira podatke o vlažnosti zraka, zračnem pritisku, temperaturi prostora, objektni temperaturi, količini svetlobe, moči magnetnega polja ter pospeških, ki delujejo nanj.



Slika 3.1: Shema tehnične rešitve. Srce predstavlja splošen senzor za zbiranje podatkov.

Vsako meritev posebej nato pošlje Android aplikaciji na mobilni napravi. Ta meritve prikazuje na zaslonu v obliki grafa v realnem času, v ozadju pa jih preko lokalnega WiFi omrežja posreduje strežniku na računalniku. Strežnik podatke prestreže in shrani v podatkovno bazo SQLite.

3.2 Opis rešitve

3.2.1 Uporaba naprave SensorTag

Posebne nastavitve na napravi SensorTag niso potrebne. Ob zagonu mobilne aplikacije ga je potrebno le vklopiti, da se mobilna aplikacija nanj lahko poveže. SensorTag vklopimo s pritiskom na gumb, ki se nahaja na levi strani naprave. Ob vklopu se zažene sistemska programska oprema naprave, ki približno dve minuti oddaja signal. Po dveh minutah sistem preide v način delovanja, ki signal oddaja le po potrebi (na primer ob pošiljanju pridobljenih podatkov), saj tak način optimizira porabo energije.

Občasno se pojavijo težave z delovanjem 9-osnega senzorja za zaznavanje gibanja, ki vključuje pospeškometer, giroskop in merilec magnetnega polja. Takrat se meritve tega senzorja prenehajo in podatkov ni več mogoče pridobiti. Težavo rešimo tako, da iz naprave za kratek čas odstranimo baterijo, ter jo ponovno prižgemo.

3.2.2 Mobilna aplikacija

Mobilna aplikacija se na Android telefonih nahaja pod imenom *BLE SensorTag*. Zgrajena je na vzorčni aplikaciji, ki je dostopna na spletni strani Texas Instruments, odločili pa smo se za Anroid aplikacijo, saj smo jo imeli v mislih že od prej. Pred uporabo jo je treba vklopiti. Takrat mora biti vklopljen tudi SensorTag, na katerem ob oddajanju vsako sekundo utripne zelena lučka. Če mobilni telefon nima vklopljenega Bluetootha, nas aplikacija zaprosi za dovoljenje, da ga vklopi, če pa ugotovi, da naš telefon tehnologije BLE ne podpira, nam to sporoči in aplikacije ne moremo uporabljati.

Ko so vklopljeni tako SensorTag kot tudi aplikacija in Bluetooth, na dnu zaslona pritisnemo na gumb z napisom *Scan*, ki poišče bližnje BLE naprave v načinu oddajanja in jih prikaže na zaslonu v obliki seznama. Pri želeni napravi izberemo gumb z napisom *Connect*, ki začne povezavo.

Ko se povezava uspešno vzpostavi se nam na zaslonu prikaže seznam pridobljenih profilov naprave SensorTag (glej sliko 2.5). Za nas uporabniški vmesnik ni bil pretirano pomemben, saj je bil naš končni cilj pridobitev podatkov na računalnik, zato ga nismo veliko spreminjali. Na seznamu se prikažejo profili za meritev vlažnosti zraka, zračnega pritiska, temperature prostora, temperature objekta, količine svetlobe in 9-osnega senzorja za zaznavanje gibanja. Na začetku je seznam vseboval še nekaj drugih polj, ki pa smo jih, ker niso bile povezane z našo implementacijo, odstranili. Prikazane so le meritve, ki jih kasneje želimo imeti na računalniku.

Za nas pomembni dogodki se dogajajo v ozadju aplikacije, kar je podrobneje opisano v razdelku 3.3.

3.3 Povezovanje naprave SensorTag na mobilno napravo

SensorTag ima vlogi strežnika in sužnja, kar kot že omenjeno v razdelku 2.4 pomeni, da s podatki razpolaga in jih pošilja, mobilna naprava pa ima vlogi odjemalca in gospodarja, torej podatke želi imeti in po njih povprašuje. Napravi si podatke izmenjujeta preko protokola BLE.

Pred iskanjem BLE oddajnikov je najprej potrebno poskrbeti, da aplikacija BLE podpira, kar storimo v manifest datoteki, kjer moramo dodati dovoljenji `BLUETOOTH` in `BLUETOOTH_ADMIN`. Za edino vstopno točko v celoten sistem preko Bluetootha skrbi `BluetoothAdapter`, ki ga pridobimo preko metode `getAdapter()`, pripadajoče razredu `BluetoothManager`. Brez njega delo s protokolom Bluetooth ni mogoče. Naprave iščemo z adapterjem in klicem `startLeScan()`, ki za svoje delovanje potrebuje implementacijo `LeScanCallback`. V tem koraku je treba pri Android sistemih, ki imajo verzijo višjo od 5.0, v manifest datoteko dodati dovoljenje `ACCESS_FINE_LOCATION`. Ko napravo najdemo v metodi `onLeScan`, s čimer dobimo objekt tipa `BluetoothDevice`, se povežemo na GATT strežnik (`BluetoothGatt`), ki ga nato uporabljamo, da lahko izvajamo odjemalske GATT operacije. Ponovno je potrebna implementacija klica `BluetoothGattCallback`, v katerem moramo poskrbeti za to, kaj se bo zgodilo ob uspešni povezavi v metodi `onConnectionStateChanged()`, za iskanje storitev v metodi `onServicesDiscovered()` in za dogodke ob prejemu nove meritve v metodi `onCharacteristicChanged()`. Vse tri metode se kličejo avtomatsko ob dogodkih, ki jih opisujejo njihova imena.

Ko smo pridobili vse storitve, ki jih naprava ponuja s klicem `getServices()`, ter nji-

hove karakteristike s klicem `getCharacteristics()`, moramo strežniški napravi omogočiti pošiljanje notifikacij za vsako karakteristiko. To storimo tako, da deskriptorju vsake karakteristike vrednost nastavimo na `ENABLE_NOTIFICATION_VALUE`.

Zdaj strežniška naprava ob pošiljanju novih podatkov odjemalca obvesti, da je podatke poslala, ta pa jih v metodi `onCharacteristicChanged()` lahko prebere s pomočjo objekta `BluetoothGattCharacteristic` in klicem `getValue()`.

3.4 Pošiljanje podatkov iz mobilne naprave na računalnik

Frekvenco oddajanja posameznih podatkov določimo v mobilni aplikaciji. V našem primeru smo oddajanje nastavili na vsaki dve sekundi, sistemska programska oprema pa sicer čas vzorčenja omogoča v mejah od 100 (pri infrardeči in ambientalni temperaturi 300) do 2550ms z 10 ms natančnostjo.

Ker smo hoteli imeti podatke, ki smo jih prebrali iz senzorja na koncu na podatkovni bazi na računalniku, jih je bilo potrebno tja poslati. V aplikaciji smo podatke zapakirali v JSON format in jih preko lokalnega omrežja s klicem Http POST poslali na preprost strežnik. Ta je podatke sprejel, jih iz JSON formata preoblikoval nazaj v objekt in porinil v SQLite podatkovno bazo.

3.4.1 Razreda HttpUrlConnection in AsyncTask

Naš strežnik ob vklopu teče na svojem mrežnem URL naslovu. Da smo nanj lahko poslali podatke, smo uporabili razred `HttpUrlConnection` [14] ter kot način pošiljanja izbrali HTTP-jev klic POST. Pošiljanje poteka po naslednjih korakih:

1. nastavljanje povezave na mrežni URL naslov s klicem `URL.openConnection()`,
2. nastavljanje opisnih podatkov in načina pošiljanja, pri čemer moramo, če bo imel HTTP klic telo, obvezno nastaviti `setDoOutput(true)`,
3. odpiranje podatkovnega toka z URL povezavo s klicem `getOutputStream()` in pisanje podatkov,
4. sprejem odgovora strežnika z uporabo metode `getInputStream()`,
5. sprostitev uporabljenih virov z zapiranjem povezave s klicem `disconnect()`.

Android aplikacije privzeto tečejo na eni niti. Če na njej želimo zagnati operacije, ki tečejo dalj časa, npr. klic na strežnik ali zapisovanje v datoteke, aplikacija med čakanjem na odgovor zamrzne in se je ne da uporabljati. Po predpisih Androida se daljših operacij na glavni niti ne sme izvajati, zato mora pošiljanje na strežnik potekati *asinhrono* in *v ozadju*. To nam omogoča razred `AsyncTask` s katerim razširimo razred, v katerem želimo pošiljati podatke. Pošiljanje implementiramo v metodi `doInBackgroud()`. Po klicu se rezultati prikažejo na glavni niti.

3.4.2 Strežnik

Na računalniku smo vzpostavili preprost strežnik v jeziku C#. Njegova naloga je, da ob zagonu, če še ne obstaja, naredi novo SQLite podatkovno bazo, nato pa čaka na HTTP POST klice. Ko klice sprejme, vsakega obdela na svoji niti. Podatke iz formata JSON preoblikuje v objekt in jih zapiše v novo vrstico v ustrezni tabeli podatkovne baze, hkrati pa jih prikaže tudi v majhnem oknu, da uporabnik vidi, da se je nekaj zgodilo. Primer izpisa v prikaznem oknu je prikazan na sliki 3.2. Ob prejemu podatkov vrne odgovor, v katerem sporoči ali je klic uspel in katere podatke je prejel. Strežnik na HTTP klice posluša na vratih 8082, ki jih je pred uporabo potrebno odpreti v požarnem zidu računalnika.

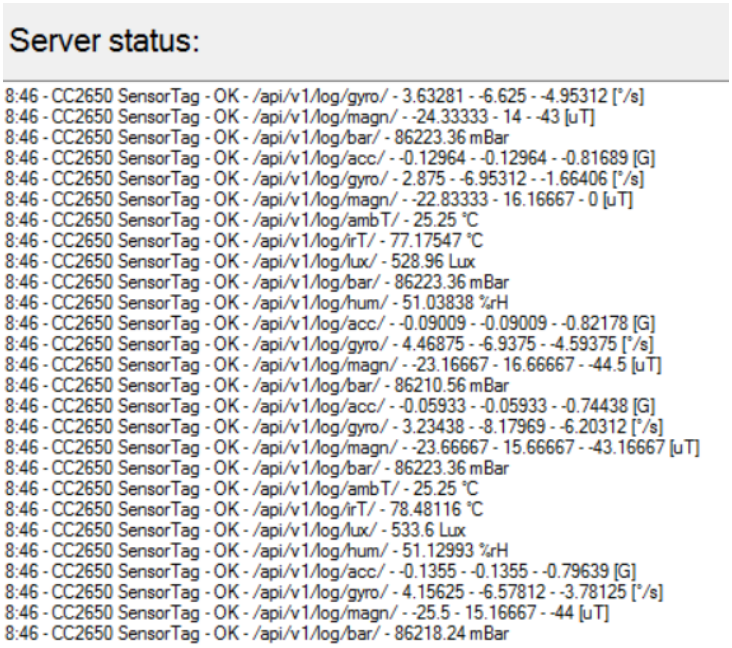
3.5 Podatkovna baza

Za shranjevanje podatkov smo uporabili SQLite podatkovno bazo. Za vsak ločen profil SensorTaga se podatki shranjujejo v svojo tabelo, ki smo jo ustvarili s klicem `CREATE TABLE`.

Klici s katerimi smo ustvarili tabele za hranjenje meritev so:

- za vlago v zraku

```
CREATE TABLE HumidityData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                    DATETIME,
        deviceTag                       TEXT,
        humidity                        REAL,
        unit                            TEXT
);
```



Slika 3.2: Prikazno okno strežnika.

- za barometer

```
CREATE TABLE BarometerData (
        _id             INTEGER PRIMARY KEY,
        dateTime                DATETIME,
        deviceTag                   TEXT,
        barometer                   REAL,
        unit                        TEXT
);
```

- za senzor ambientalne svetlobe

```
CREATE TABLE LuxometerData (
        _id             INTEGER PRIMARY KEY,
        dateTime                DATETIME,
        deviceTag                   TEXT,
        luxometer                   REAL,
        unit                        TEXT
);
```

- za infrardeči senzor temperature

```
CREATE TABLE IRTemperatureData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                        DATETIME,
        deviceTag                           TEXT,
        irTemperature                       REAL,
        unit                                TEXT
);
```

- za senzor ambientalne temperature

```
CREATE TABLE AmbientTemperatureData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                        DATETIME,
        deviceTag                           TEXT,
        ambientTemperature                  REAL,
        unit                                TEXT
);
```

- za pospeškometer

```
CREATE TABLE AccelometerData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                        DATETIME,
        deviceTag                           TEXT,
        accX                                REAL,
        accY                                REAL,
        accZ                                REAL,
        unit                                TEXT
);
```

- za giroskop

```
CREATE TABLE GyroscopeData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                        DATETIME,
```

```
        deviceTag                       TEXT,
        gyroX                           REAL,
        gyroY                           REAL,
        gyroZ                           REAL,
        unit                            TEXT
);
```

- za merilec magnetnega polja

```
CREATE TABLE MagnetometerData (
        _id                 INTEGER PRIMARY KEY,
        dateTime                        DATETIME,
        deviceTag                       TEXT,
        magnX                           REAL,
        magnY                           REAL,
        magnZ                           REAL,
        unit                            TEXT
);
```

Tabela vsebuje podatek o času shranjevanja, imenu naprave, ki je je podatke poslala, numerični meritvi in enoti merjenja. Pri senzorju vlažnosti je ta `%rH` (relativna vlaga), senzorju zračnega pritiska `mBar`, senzorju ambientalne svetlobe `lux` (količina svetlobe), pri pospeškometru `G` (gravitacijski pospešek), giroskopu `°/s` (kotni pospešek), merilcu magnetnega polja `uT` (mikro Tesla), pri obeh temperaturnih senzorjih pa `°C` ali `°F`, odvisno od nastavitev.

Slika 3.3 prikazuje izgled tabele z meritvami senzorja vlažnosti v SQLite podatkovni bazi.

_id	dateTime	deviceTag	humidity	unit
1	2016-09-29 08:42:02.2498053	CC2650 SensorTag	48.9509391784668	%rH
2	2016-09-29 08:42:04.1257623	CC2650 SensorTag	48.9509391784668	%rH
3	2016-09-29 08:42:06.5221361	CC2650 SensorTag	48.9509391784668	%rH
4	2016-09-29 08:42:08.9625407	CC2650 SensorTag	48.8593902587891	%rH
5	2016-09-29 08:42:11.4056926	CC2650 SensorTag	48.8593902587891	%rH

Slika 3.3: Izgled tabele z meritvami senzorja vlažnosti.

4 Zaključek

Pilotno fazo, predstavljeno v uvodu naloge, smo uspešno implementirali. Na začetku smo precej časa namenili izbiri senzorske naprave, ki bi zadostila našim potrebam, bila zanimiva in hkrati prosto dostopna na trgu. Veliko težo smo namenili iskanju naprav, ki so bile označene kot *open-source*, kar pomeni, da skupaj z napravo proizvajalec izda tudi prosto dostopno kodo, ki omogoča lažje spoznavanje z delovanjem naprave in programiranje. Odločali smo se med tem kakšno vrsto naprave izbrati, saj obstaja kar nekaj različnih senzorskih naprav, od pametnih prevlek za posteljo, vložkov za čevlje in talnih ploščic, do zapestnic in drugega nakita. Predvsem zanimive so se nam zdele pametne škatlice za zdravila in prehranska dopolnila, o katerih smo veliko brali na internetu, saj so povezane tako z zdravstvom na splošno kot tudi specifično s starostniki, ki živijo v oskrbovanih stanovanjih. Mnogo starostnikov je zaradi različnih zdravstvenih težav podrejeno zapletenim in strogim režimom jemanja zdravil, kjer je pomembno da se točno določena zdravila vzame ob točno določenem času, zato bi jim take škatlice z opomniki lahko močno olajšale pravilno jemanje zdravil. Na internetu so bile pogosto omenjene tudi tablete, ki ko jih pacient zaužije to sporočijo na pametno napravo.

Žal pa na koncu nobene od pametnih škatlic za tablete nismo uspeli dobiti. Veliko jih je šele v fazi prototipiranja in testiranja, nekaj jih ni mogoče kupiti v prosti prodaji, temveč le v imenu večje farmacevtske firme in v večjih količinah, nekatere ne podpirajo odprte kode, zadnjih nekaj pa nam ni bilo cenovno dostopnih.

Tako smo se na koncu odločili, da bomo projekt izvedli z zapestnico, ki bo merila zdravstvene podatke starostnika. Med zapestnicami, ki podpirajo odprto kodo, se je na spletu največ pisalo o zapestnici AngelSensor, pa tudi spletna stran proizvajalca je izgledala zelo obetavno. Našli smo celo kar nekaj dokumentacije. Ker se prej še nihče od nas ni ukvarjal s podobnimi rečmi, nam je AngelSensor izgledal kot obetavna senzorska naprava. Po nakupu pa so se kmalu začele pojavljati težave. Zapestnica je naprej delala občasno, nato pa je popolnoma odpovedala in povezava s telefonom ni bila več mogoča. Na uradni spletni strani in v dokumentaciji nismo našli nobene rešitve, ki bi nam zapestnico pomagala ponovno usposobiti. Po nekaj več iskanja smo odkrili veliko drugih nezadovoljnih kupcev s podobnimi težavami, proizvajalci pa na spletno pošto niso odgovarjali. Zato smo AngelSensor odložili in poiskali nov senzor.

Tokrat smo bili pozorni na to, da bo naša senzorska naprava res delovala, zato smo izbrali SensorTag podjetja Texas Instruments, ki se je izkazala kot dokaj zanesljiva in prilagodljiva platforma. Kljub priznanemu proizvajalcu pa tudi SensorTag ne dela vedno brez težav. 9-osni senzor za zazanavaje gibanja se po poslanih nekaj meritvah pogosto izklopi in meritve neha pošiljati. Ponovni vklop naprave, mobilne aplikacije in ponovna povezava ne pomagata. Da senzor spet usposobimo, je potrebno za kratek čas odstraniti baterijo.

Glede na naše težave z izbirami senzorske naprave lahko vidimo, da je Internet stvari, čeprav težko najdemo tehnološki članek, kjer ni omenjen in opevan v oblake, pravzaprav šele v povojih. Vsaj kar se tiče prostega uporabniškega trga. Na nekoliko drugem nivoju ga obravnavajo velike tehnološke firme kot so IBM, Google ipd., ki pa imajo že razvite končne in delujoče rešitve in sisteme.

Sprva smo želeli s senzorsko napravo zbirati zdravstvene podatke kot sta srčni utrip in temperatura kože o osebi, ki bi jo nosila, SensorTag pa s svojimi senzorji iz okolja pobira podatke o temperaturi, svetlosti in drugih lastnostih prostora. Ker pa smo svoj projekt že na začetku postavili kot pilotni projekt zbiranja podatkov v celotnem varovanem

stanovanju nas to ni preveč motilo. Tudi podatki o stanju prostora namreč pripomorejo h celotnem senzorskem pokritju pametnega varovanega stanovanja in nam povedo mnogo stvari.

Na analizo podatkov se v tej fazi še nismo osredotočali, sej smo jih najprej morali znati iz SensorTaga prebrati. To smo storili z uporabo tehnologije Bluetooth Low Energy – BLE. SenzorTag meritve iz različnih senzorjev pošilja kot ločene storitve, ki so entitete v protokolu GATT, ki narekuje pravila za povezovanje naprav in komunikacijo z BLE. Določa, da se sporočila prenašajo v obliki atributov, katerih sestavni del so poleg storitev še karakteristike in deskriptorji. Sam postopek povezave SensorTaga in mobilne naprave je bil že implementiran v aplikaciji, ki jo proizvajalec prosto ponuja na spletu, zato nam je bilo v tem koraku prikrajšanega nekaj dela. Vseeno pa koda ni popolnoma sledila navodilom vzpostavljanja povezave, ki ga za BLE določa Android in ni vsebovala veliko komentarjev, zato je bilo določene dele postopka nekoliko težje razbrati. Dokumentacija je dobra in proizvajalec je na svoji spletni strani objavil celotno tabelo unikatnih identifikacijskih številk UUID vseh atributov in njihovih sestavnih delov. Z pisanjem v deskriptor CCCD (opisan v razdelku 2.4.1) smo lahko za vsak določen senzor določali ali njegove podatke želimo sprejemati ali ne.

Ko smo podatke prejeli na telefon, jih je bilo potrebno posredovati naprej do podatkovne baze, kar smo storili z vmesnim korakom strežnika. Napisali smo preprost strežnik, ki je skrbel za ustvarjanje podatkovne baze, če ta še ni obstajala, za prejemanje podatkov in njihovo shranjevanje v bazo. Za podatkovno bazo smo izbrali SQLite.

Podatke smo poslali preko lokalnega omrežja asinhrono in v ozadju aplikacije s HTTP POST klicem v obliki JSON na mrežni naslov URL, na katerem je tekel strežnik. Ta je podatek iz JSON oblike z deserializacijo (ang. *deserialize*) preoblikoval v objekt, ki ga je porinil v tabelo na podatkovni bazi, ki ji je meritev pripadala. Ob uspehu se je rezultat še izpisal v manjše okno na zaslonu, kjer se je zapisalo kateri podatek se je zapisal v katero tabelo.

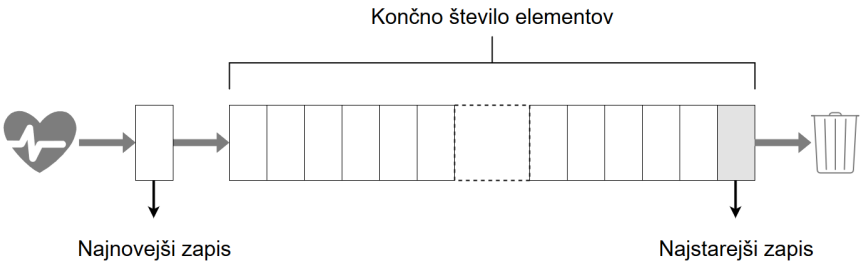
Podatke v naši nalogi sicer le prebiramo iz okolja in shranjujemo v bazo, ter jih ne analiziramo. Iz njih pa bi bilo v nadaljevanju mogoče pridobiti mnogo uporabnih opazovanj. Z merjenjem sobne temperturature bi lahko ugotovili ali ima oskrbovanec poleti dovolj hladno in pozimi dovolj toplo. Ob nenadnem padcu temperature pozimi bi lahko sklepali,

da je starostnik odprl okno in ga mogoče celo pozabil zapreti. Ob obratni situaciji, kjer bi se v klimatiziranem prostoru temperatura naenkrat močno dvignila, bi lahko avtomatsko zmanjšali hlajenje. S podatki iz senzorja količine svetlobe bi lahko ugotavljali ob katerih urah starostnik hodi spat ugotovili in z analizo ugotovili ali ima težave s spanjem, ki bi lahko bile povezane s kakšnim zdravstvenim sindromom. Pozimi gretje osuši zrak, ki je mogoč vpliv na spremembo počutja starostnika, zato bi mu lahko svetovali, naj navlaži zrak. SensorTag ima sicer nastavek za zapestni trak ampak je nekoliko velik, zato se nam kot senzor gibanja osebe ne zdi pretirano uporaben. Lahko pa bi tudi bil.

Našo podatkovno bazo bi v naslednji fazi morali nekoliko nadgraditi. Trenutno se podatki vanjo beležijo brez ozira na velikost podatkovne baze. Pri manjši količini podatkov to še ni problem, če pa bi SenzorTag pustili teči daljše časovno obdobje, bi podatkovna baza lahko prerasla v neobvladnjivo velikost. Postala bi rigidna in težje obvladljiva. Najprej bi morali razmisliti o daljših časovnih razmikih branja podatkov iz okolja s SensorTagom. Okoljskih meritev znotraj varovanega stanovanja verjetno ni potrebno prebirati vsaki dve sekundi kot to počnemo zdaj, temveč bi zadostovale meritve vsakih nekaj minut. Strojna oprema frekvenco merjenja trenutno omejuje na 100-2550ms z natančnostjo 10ms, a SensorTag z dodatnim priključkom omogoča tudi programiranje strojne opreme, kjer bi meje frekvenc merjenja lahko spremenili.

Poleg tega pa tako velike količine podatkov niso uporabne niti za analizo. Določiti bi morali kako stari podatki nam informacijsko še lahko koristijo, dlje pa jih ne bi hranili. Našo podatkovno bazo bi v tem primeru lahko preoblikovali v krožno ali RR (ang. *Round-Robin*) podatkovno bazo. Krožne baze delujejo tako, da jim omejimo velikost oziroma določimo največje število vrstic. Ko se posamezna tabela napolni, za novo vrstico ni več prostora. Da zanjo dobimo prostor iz tabele izbrišemo najstarejši podatek in na njegovo mesto zapišemo novega. Za takšno bazo rečemo, da je krožna, saj najnovejši podatek vedno zamenja najstarejšega in menjave podatkov tako potekajo v krogu. Način delovanja krožne podatkovne baze je prikazan na sliki 4.1.

Naslednja nadgraditev naše podatkovne baze bi bilo lahko preblikovanje formata posamezne tabele. Trenutno vanje shranjujemo podatke v obliki, kot jih iz naprave prebiramo in brez posebno premišljenega formata. Poleg meritev senzorja smo dodali še podatke, ki jo dodatno opisujejo. To so čas meritve, ime naprave in merska enota. Naša ideja



Slika 4.1: Delovanje krožne podatkovne baze. Srce simbolično prikazuje senzorsko napravo, ki je vir podatkov. Shema je bila narisana s pomočjo programa Flowchart Maker [10].

pa je bila, da bi računalnik oziroma koncentrator, kjer bi zbirali podatke meritev vseh senzorskih naprav v stanovanju, lahko deloval kot povezovalna točka med pridobljenimi meritvami in zunanjim svetom.

Do podatkov bi preko spleta lahko dostopalo zdravstveno osebje, ki bi nadzorovalo starostnikovo vedenje in počutje; sorodniki, ki bi preko preprostega uporabniškega vmesnika lahko preverili, da je z njihovim bližnjim še vse dobro in jim ne bi bilo treba pretirano skrbeti; ob kritičnih spremembah v stanovanju ali zdravstvenem stanju starostnika pa bi se lahko celo avtomatsko poslali alarmi prvi pomoči ali gasilcem. Meritve vitalnih znakov bi se lahko v določenih intervalih pošiljale v starostnikov zdravstveni karton, da bi imel pregled nad njimi tudi njihov osebni zdravnik.

Če želimo brez posebnih prilagajanj med seboj povezati različne sisteme je dobro, da smo univerzalni. Obstaja nekaj standardov, ki določajo v točno kakšnih oblikah morajo biti shranjeni zdravstveni podatki pacientov, da so prosto prenosljivi med različnimi zdravstvenimi ustanovami. Taki standardi so naprimer CEN 13606, HL7, in openEHR, ki bi jih lahko podrobneje raziskali in podatkovno bazo prilagodili njihovem formatu.

Literatura

[1] World Population Ageing: 1950-2050, http://www.un.org/esa/population/publications/worldageing19502050/ (September 2016).

[2] Angel Sensor, http://angelsensor.com/ (August 2016).

[3] Angel Sensor - Research, http://angelsensor.com/research/ (August 2016).

[4] Sensor Tag CC2650, http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html (September 2016).

[5] IAR Embedded Workbench, https://www.iar.com/iar-embedded-workbench/ (September 2016).

[6] Code Composer Studio IDE, www.ti.com/tool/ccstudio (September 2016).

[7] CC2650 SensorTag User's Guide, http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide#Gatt_Server (September 2016).

[8] Android (operating system), https://en.wikipedia.org/wiki/Android_(operating_system) (August 2016).

[9] Meet Android Studio, https://developer.android.com/studio/intro/index.html (August 2016).

[10] Flowchart maker, https://www.draw.io/ (September 2016).

[11] BLE storitev meritve srčnega utripa, https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.heart_rate.xml (August 2016).

[12] Getting started with BLE, https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/ch04.html (August 2016).

[13] About SQLite, http://www.sqlite.org/about.html (September 2016).

[14] HttpURLConnection, https://developer.android.com/reference/java/net/HttpURLConnection.html (September 2016).

[15] H. Aloulou, M. Mokhtari, T. Tiberghien, J. Biswas, C. Phua, J. H. Heneth Lin, P. Yap, Deployment of assistive living technology in a nursing home environment: methods and lessons learned, BMC medical informatics and decision making (13) (2013) 42–.

[16] M. T. Ferreira, S. M. Matsudo, M. C. Ribeiro, L. R. Ramos, Health-related factors correlate with behavior trends in physical activity level in old age: longitudinal results from a population in são paulo, brazil, BMC Public Health 10 (1) (2010) 1–10.

[17] T. Instruments, Cc2650 simplelink developer's guide.

[18] E. Jenko, Uporaba senzorjev gibanja v oskrbovanem stanovanju, MSc thesis (2015).

[19] J. Lloret, A. Canovas, S. Sendra, L. Parra, A smart communication architecture for ambient assisted living, IEEE Communications Magazine 53 (1) (2015) 26–33.

[20] M. Memon, S. R. Wagner, C. F. Pederesen, F. H. A. Beevi, F. O. Hansen, Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes, Sensors (Basel, Switzerland) 14 (2014) 4312–4341.

[21] N. Zouba, F. Bremond, M. Thonnat, An Activity Monitoring System for Real Elderly at Home: Validation Study, in: 7th IEEE International Conference on Advanced Video and Signal-Based Surveillance AVSS10, 2010.